I. Các câu hỏi từ 1 đến 5

1. Mô tả bằng dạng Bảng và ngôn ngữ tự nhiên các chức năng

1.1. Mô tả bằng ngôn ngữ tự nhiên

1.1.1. User Module

- Admin có thể thêm một customer mới
- Admin có thể xem danh sách chi tiết các customer
- Admin có thể xóa một customer
- Customer có thể đăng ký tài khoản hệ thống
- Customer có thể chỉnh sửa và cập nhật thông tin của mình
- Customer có thể đổi mật khẩu của mình
- Customer có thể thực hiện xác thực danh tính của mình

1.1.2. Manager Module

- Admin có thể thực hiện xác thực danh tính của minh
- Admin có thể chỉnh sửa và cập nhật thông tin của mình
- Admin có thể đổi mật khẩu của mình

1.1.3. Book Module

- Admin có thể xem danh sách chi tiết các Book
- Admin có thể thêm mới một Book
- Admin có thể sửa/xóa một Book
- Customer có thể xem danh sách chi tiết các Book

1.1.4. Mobile Module

- - Admin có thể xem danh sách chi tiết các Mobile
- Admin có thể thêm mới một Mobile
- Admin có thể sửa/xóa một Mobile
- Customer có thể xem danh sách chi tiết các Mobile

1.1.5. Clothes Module

- Admin có thể xem danh sách chi tiết các Clothes
- Admin có thể thêm mới một Clothes
- Admin có thể sửa/xóa một Clothes

- Customer có thể xem danh sách chi tiết các Clothes

1.1.6. Search Module

- Admin có thể tìm kiếm sản phẩm (Book, Clothes, Mobile) bằng keyword

- Customer có thể sản phẩm (Book, Clothes, Mobile) bằng keyword, voice

1.1.7. Cart Module

- Customer có thể tạo và thêm một sản phẩm vào Cart

- Customer có thể sửa/xóa sản phẩm trong Cart

1.1.8. Order Module

- Admin có thể duyệt (thêm) đơn hàng mới

- Admin có thể xem danh sách chi tiết đơn hàng

- Admin có thể chỉnh sửa và cập nhật đơn hàng

- Admin có thể hủy (xóa) đơn hàng

- Customer có thể đặt hàng

- Customer có thể xem thông tin các đơn hàng của mình

- Customer có thể hủy đơn hàng nếu đơn hàng chưa được duyệt

1.1.9. Payment Module

- Bank có thể xử lý thanh toán

- Customer có thể xem các phương thức thanh toán mình đã dùng

- Customer có thể xem lịch sử thanh toán

- Admin có thể xem lịch sử giao dịch

1.1.10. Shipment Module

- Admin có thể thêm thông tin về hãng vận chuyển

- Admin có thể sửa/xóa thông tin về hãng vận chuyển

- Admin có thể xem danh sách chi tiết về hãng vận chuyển

- Customer có thể xem thông tin về hãng vận chuyển

- Customer có thể xem thông tin địa chỉ vận chuyển

- Customer có thể thêm thông tin địa chỉ vận chuyển mới

- Customer có thể sửa thông tin địa chỉ vận chuyển

- Customer có thể xóa thông tin địa chỉ vận chuyển

Dưới đây là bảng chi tiết về các chức năng của từng module:

| Actor | Functionalities | Detail Description
|
|---|---|---
| Customer | Cập nhật Profile | - Customer click “Quản lý thông tin cá nhân” → Hệ thống hiện giao diện quản lý thông tin của người dùng. |
| | | - Nếu Customer click “Sửa thông tin” → Hệ thống hiện giao diện sửa thông tin với các thông tin có thể sửa → Customer sửa thông tin muốn sửa và click lưu → Hệ thống cập nhật thông tin Customer. |
| | | - Nếu Customer click “Đổi mật khẩu” → Hệ thống hiện giao diện đổi mật khẩu với các ô nhập thông tin cần thiết → Customer nhập mật khẩu cũ, nhập khẩu mới rồi click đổi → Hệ thống xác minh thông tin và cập nhật mật khẩu cho Customer. |
| Đăng nhập | Customer nhập thông tin đăng nhập trên giao diện đăng nhập → Hệ thống xác minh và thông báo thông tin đăng nhập cho người dùng. |
| Đăng ký | Khách click “Đăng ký” → Hệ thống hiện giao diện đăng ký với các trường thông tin cần thiết → Khách nhập các thông tin cần thiết → Hệ thống xác minh thông tin và thông báo thông tin đăng ký cho Khách. |
| Tìm kiếm sản phẩm | - Nếu Customer nhập keyword và click tìm → Hệ thống hiện danh sách các sản phẩm chứa keyword. → Customer click chọn 1 sản phẩm → Hệ thống hiện giao diện thông tin chi tiết sản phẩm đã chọn. |
| | - Nếu Customer click micro và nói keyword → Hệ thống hiện danh sách các sản phẩm chứa keyword → Customer click chọn 1 sản phẩm → Hệ thống hiện giao diện thông tin chi tiết sản phẩm đã chọn. |
| Quản lý giỏ hàng | - Ở giao diện chi tiết sản phẩm, Customer click thêm vào giỏ hàng → Hệ thống thêm sản phẩm đó vào giỏ hàng và chuyển qua giao diện giỏ hàng với thông tin sản phẩm đã thêm, thông tin sản phẩm thêm trước đó và tổng giá. |
| | - Ở giao diện giỏ hàng, Customer click xóa một sản phẩm khỏi giỏ → Hệ thống hiện giao diện giỏ hàng với thông tin giỏ hàng mất đi một sản phẩm. |
| | - Ở giao diện giỏ hàng, Customer click sửa số lượng một sản phẩm → Hệ thống hiện giao diện giỏ hàng với thông tin về số lượng của sản phẩm và tổng giá đã được sửa. |

| Đặt hàng | - Ở giao diện giỏ hàng, Customer ấn chọn một số sản phẩm muốn đặt và click đặt hàng → Hệ thống chuyển đến giao diện thanh toán. |

| | - Customer click chọn xem thông tin đơn hàng → Hệ thống hiện giao diện danh sách các đơn hàng → Customer click chọn một đơn hàng → Hệ thống hiện giao diện xem thông tin chi tiết đơn hàng. Nếu Customer click hủy đơn hàng (Nếu có) thì hệ thống xác minh và thông báo thông tin hủy đơn hàng. |

| Thanh toán | - Ở giao diện thanh toán, Customer nhập thông tin thanh toán và click thanh toán → Hệ thống xác minh và thông báo kết quả thanh toán cho Customer. |

| | - Customer click chọn xem lịch sử thanh toán → Hệ thống hiện giao diện xem lịch sử thanh toán của Customer. |

| Quản lý thông tin địa chỉ vận chuyển | - Customer click “Quản lý thông tin địa chỉ vận chuyển” → Hệ thống hiện giao diện quản lý thông tin địa chỉ vận chuyển. |

| | - Nếu Customer click thêm mới thông tin địa chỉ vận chuyển → Hệ thống hiện giao diện thêm mới địa chỉ vận chuyển → Customer nhập địa chỉ mới và click lưu → Hệ thống thêm địa chỉ vận chuyển mới của Customer. |

| | - Nếu Customer click sửa thông tin địa chỉ vận chuyển → Hệ thống hiện giao diện sửa địa chỉ vận chuyển → Customer sửa các thông tin về địa chỉ và click lưu → Hệ thống cập nhật thông tin địa chỉ vận chuyển của Customer. |

| | - Nếu Customer click xóa thông tin địa chỉ vận chuyển → Hệ thống xác minh và thông báo thông tin xóa địa chỉ vận chuyển cho Customer. |

| Admin | Đăng nhập | - Admin nhập thông tin đăng nhập trên giao diện đăng nhập của admin → Hệ thống xác minh và thông báo kết quả đăng nhập cho Admin. |

| | Quản lý thông tin Customer | - Admin click “Quản lý thông tin người dùng” → Hệ thống hiện giao diện quản lý thông tin người dùng. Nếu Admin click chọn một người dùng → Hệ thống hiện giao diện xem chi tiết thông tin cơ bản người dùng đó. Nếu Admin click chọn xóa một người dùng → Hệ thống xác minh và thông báo kết quả xóa cho Admin. Nếu Admin click thêm mới một người dùng → Hệ thống hiện giao diện thêm mới người dùng với các trường cần thiết → Admin nhập thông tin người dùng cần thêm và click lưu → Hệ thống xác minh và thông báo kết quả thêm người dùng cho Admin. |

|

| Quản lý thông tin sản phẩm | - Admin click “Quản lý thông tin sản phẩm” → Hệ thống hiện giao diện quản lý thông tin sản phẩm. Nếu Admin click chọn một sản phẩm → Hệ thống hiện giao diện xem chi tiết sản phẩm đó. Nếu Admin click chọn xóa một sản phẩm → Hệ thống xác minh và thông báo kết quả xóa sản phẩm cho Admin. Nếu Admin click thêm một sản phẩm mới → Hệ thống hiện giao diện thêm sản phẩm mới → Admin nhập thông tin sản phẩm mới và click lưu → Hệ thống xác minh và thông báo kết quả thêm mới sản phẩm cho Admin. Nếu Admin click sửa một sản phẩm → Hệ thống hiện giao diện sửa thông tin sản phẩm → Admin sửa thông tin sản phẩm và click lưu → Hệ thống xác minh và

thông báo kết quả cập nhật sản phẩm. Nếu Admin nhập keyword và click tìm → Hệ thống hiện giao diện với các sản phẩm chứa keyword. |

| | Cập nhật Profile Admin | - Admin click “Quản lý thông tin cá nhân” → Hệ thống hiện giao diện quản lý thông tin của quản lý. Nếu Admin click “Sửa thông tin” → Hệ thống hiện giao diện sửa thông tin với các thông tin có thể sửa → Admin sửa thông tin muốn sửa và click lưu → Hệ thống cập nhật thông tin Admin. Nếu Admin click “Đổi mật khẩu” → Hệ thống hiện giao diện đổi mật khẩu với các ô nhập thông tin cần thiết → Admin nhập mật khẩu cũ, nhập khẩu mới rồi click đổi → Hệ thống xác minh thông tin và cập nhật mật khẩu cho Admin. |

| | Quản lý thông tin đơn hàng | - Admin click “Quản lý thông tin đơn hàng” → Hệ thống hiện giao diện quản lý thông tin đơn hàng. Nếu Admin click chọn 1 đơn hàng → Hệ thống hiện giao diện xem chi tiết đơn hàng đó. Nếu Admin click chọn xem các đơn hàng chờ duyệt → Hệ thống hiện giao diện xem các đơn hàng chờ duyệt → Admin duyệt/hủy một đơn hàng → Hệ thống xác minh và thông báo kết quả cho Admin. Nếu Admin click sửa thông tin một đơn hàng → Hệ thống hiện giao diện sửa đơn hàng → Quản lý nhập các thông tin cần sửa và click lưu → Hệ thống xác minh và thông báo kết quả. Nếu Admin click hủy một đơn hàng đã duyệt → Hệ thống xác minh và thông báo kết quả. |

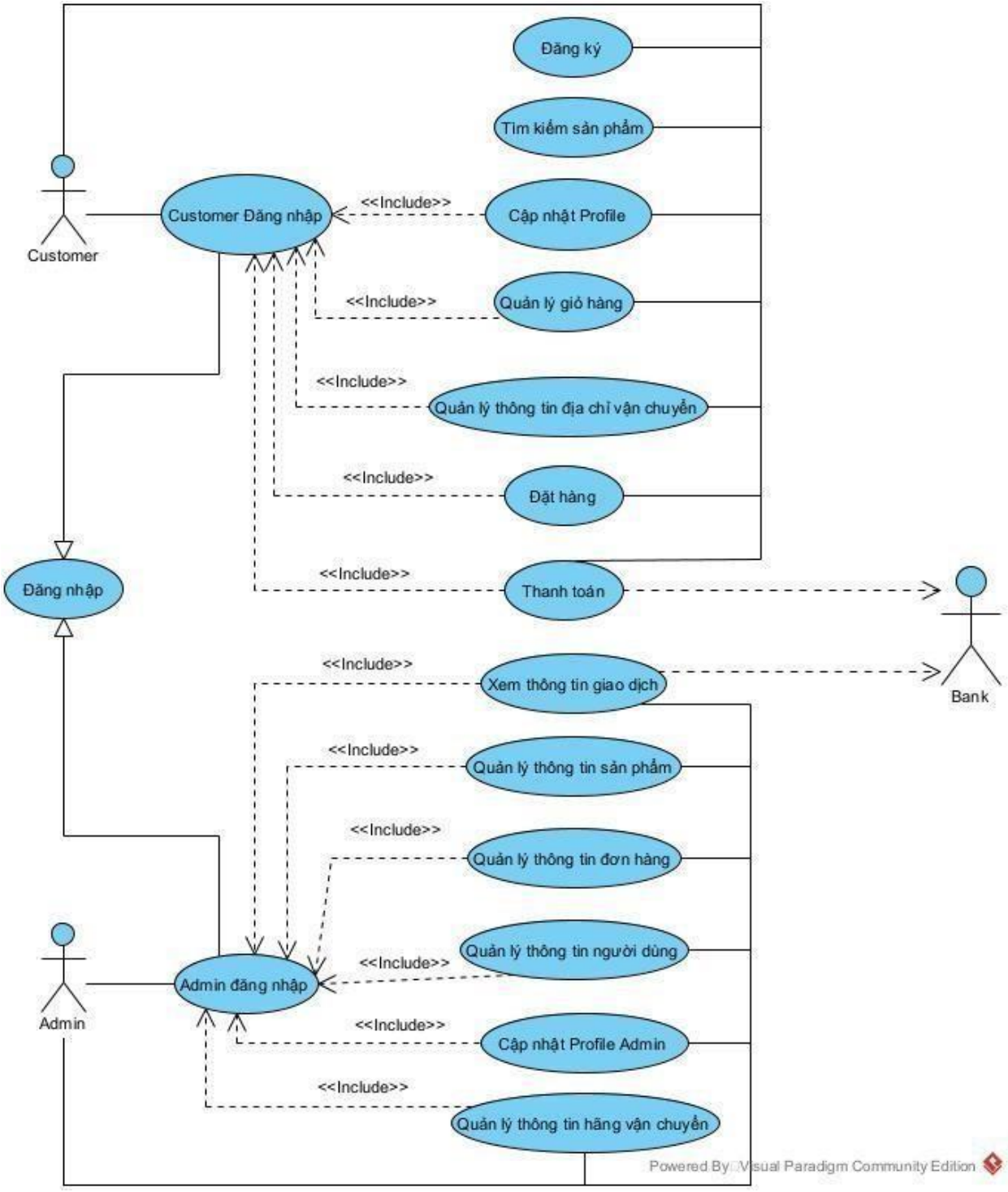
| | Xem thông tin giao dịch | - Ở giao diện chính của Admin, Admin click chọn xem lịch sử giao dịch → Hệ thống hiện thông tin các giao dịch cho Admin → Quản lý click chọn một giao dịch → Hệ thống hiện thông tin chi tiết giao dịch đó. |

| | Quản lý thông tin hãng vận chuyển | - Admin click “Quản lý thông tin hãng vận chuyển” → Hệ thống hiện giao diện xem thông tin các hãng vận chuyển. Nếu Admin click chọn một hãng vận chuyển → Hệ thống hiện giao diện xem chi tiết hãng vận chuyển đó. Nếu Admin click chọn thêm mới một hãng vận chuyển → Hệ thống hiện giao diện thêm mới hãng vận chuyển → Admin nhập thông tin hãng vận chuyển mới và click lưu → Hệ thống xác minh và thông báo kết quả cho Admin. Nếu Admin click xóa một hãng vận chuyển → Hệ thống xác minh và thông báo kết quả cho Admin. Nếu Admin click sửa một hãng vận chuyển → Hệ thống hiện giao diện sửa thông tin hãng vận chuyển → Admin sửa thông tin hãng vận chuyển và click lưu → Hệ thống xác minh và thông báo kết quả cho Admin. |

| Bank | Xử lý thanh toán | - Bank xác thực thông tin thanh toán mà Customer gửi. Nếu thông tin hợp lệ, Bank tiến hành thanh toán cho đơn hàng. Nếu thông tin không hợp lệ, Bank báo lỗi về hệ thống. |

2. Vẽ biểu đồ use case tổng quát và biểu đồ use case chi tiết cho từng chức năng dịch vụ

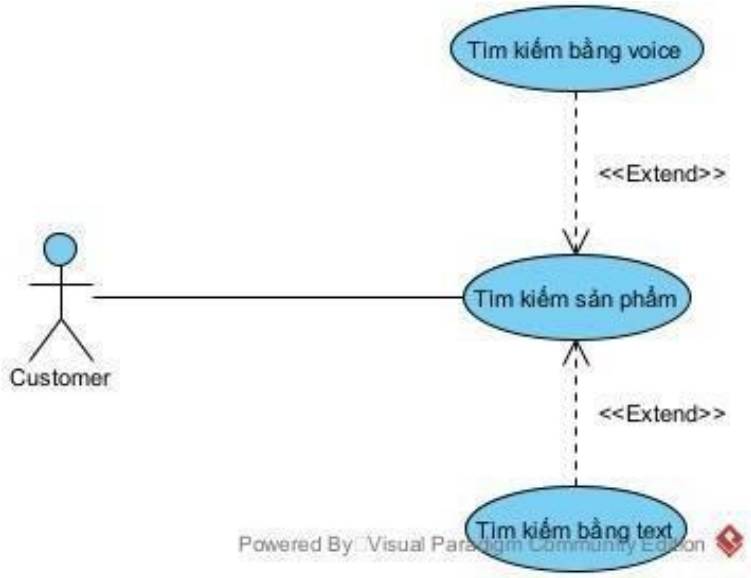
2.1. Use case tổng quát



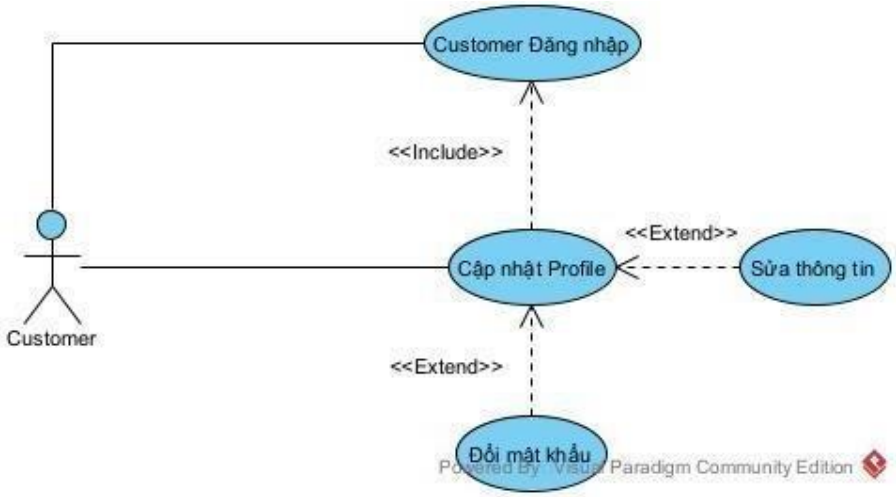
2.2. Use case chi tiết

2.2.1. Đăng ký

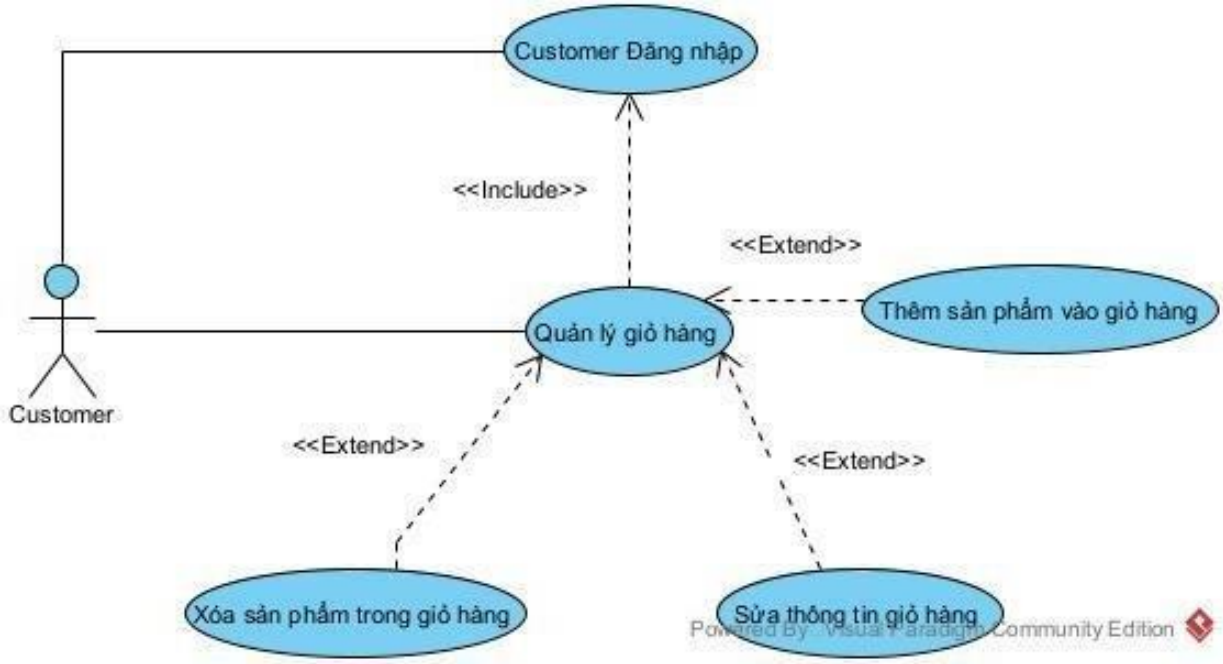
2.2.2. Tìm kiếm sản phẩm



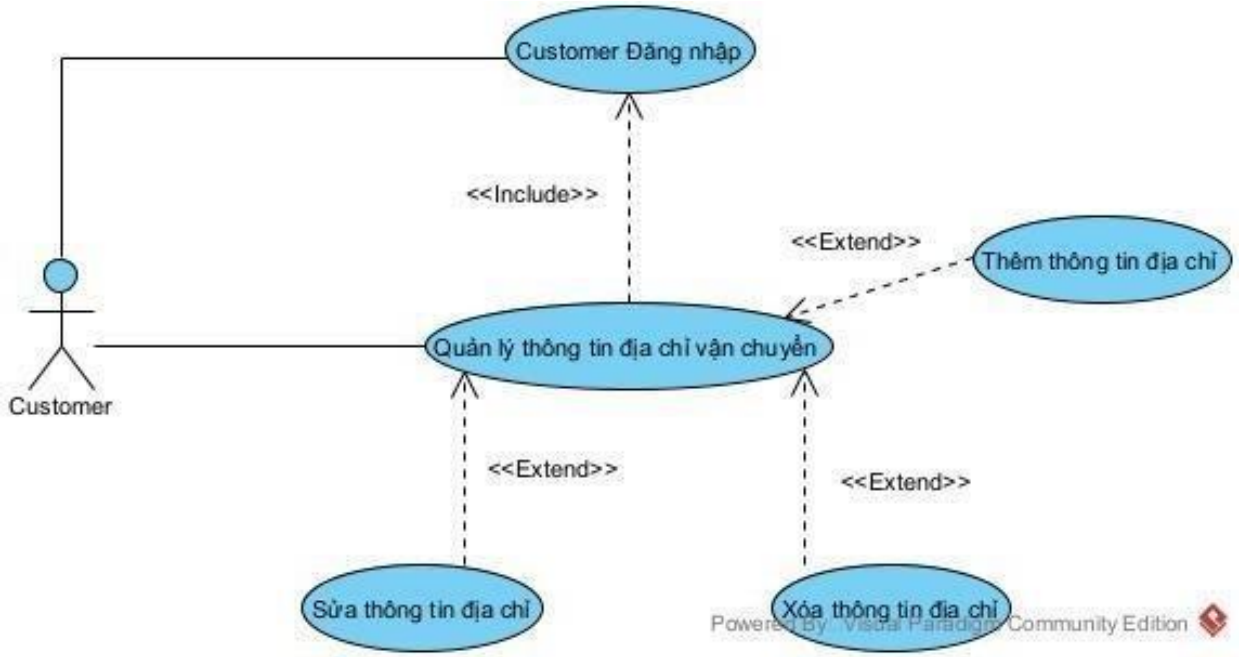
2.2.3. Cập nhật Profile



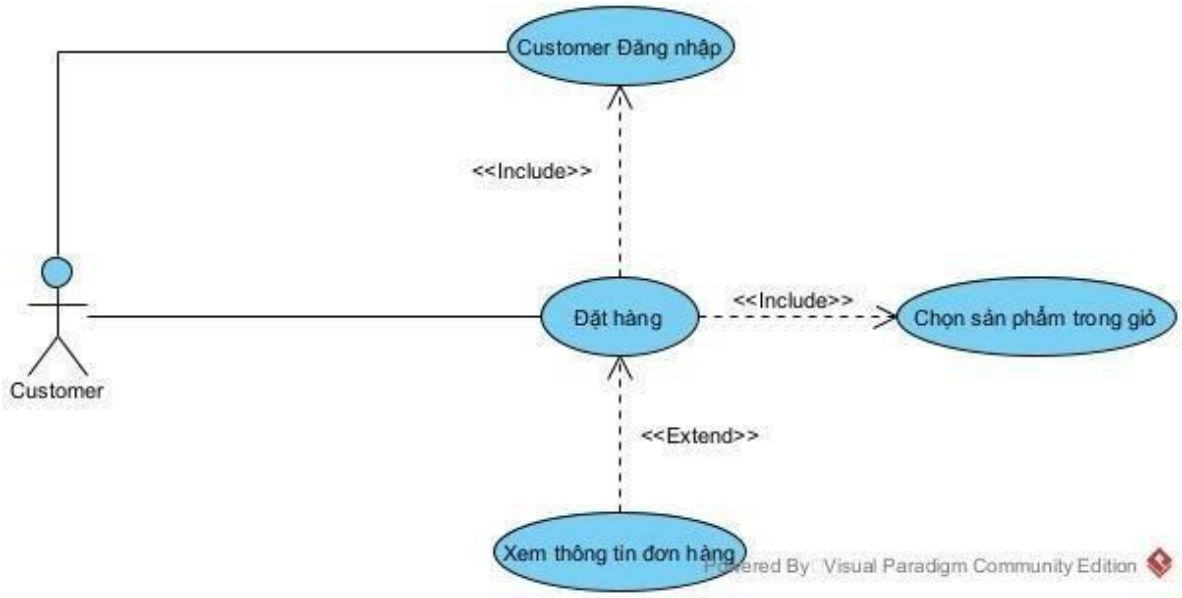
2.2.4. Quản lý giỏ hàng



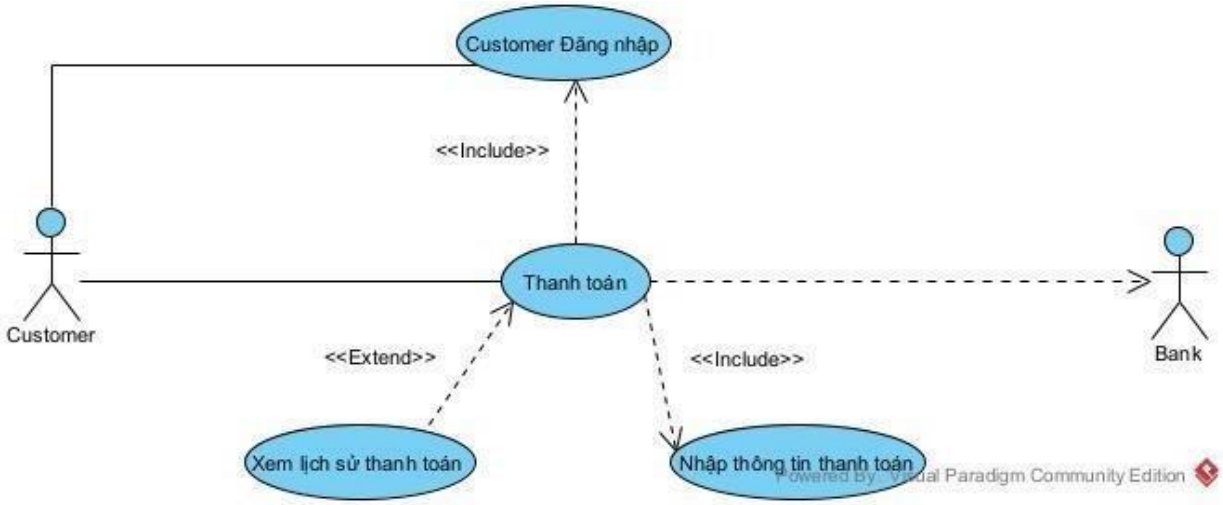
2.2.5. Quản lý thông tin địa chỉ vận chuyển



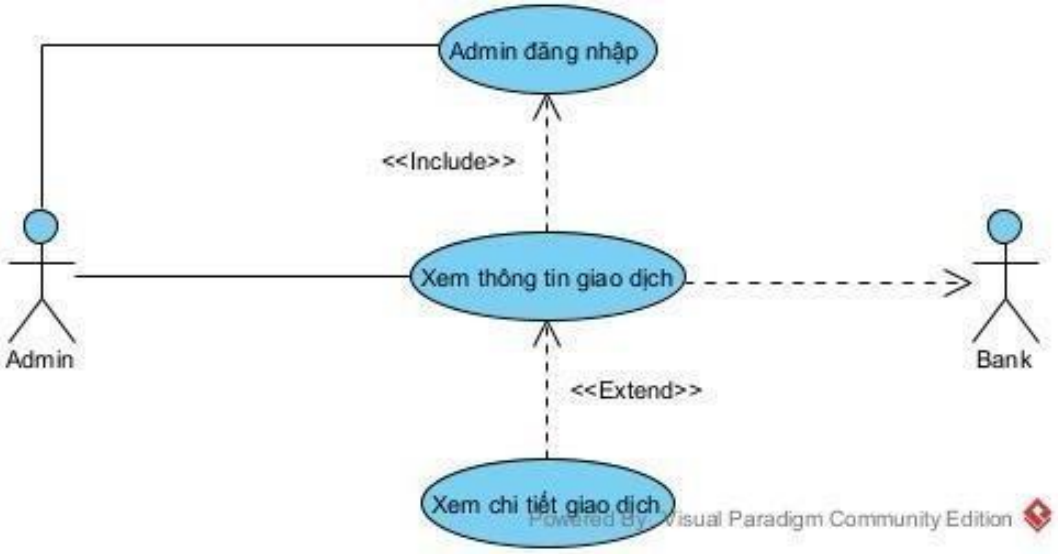
2.2.6. Đặt hàng



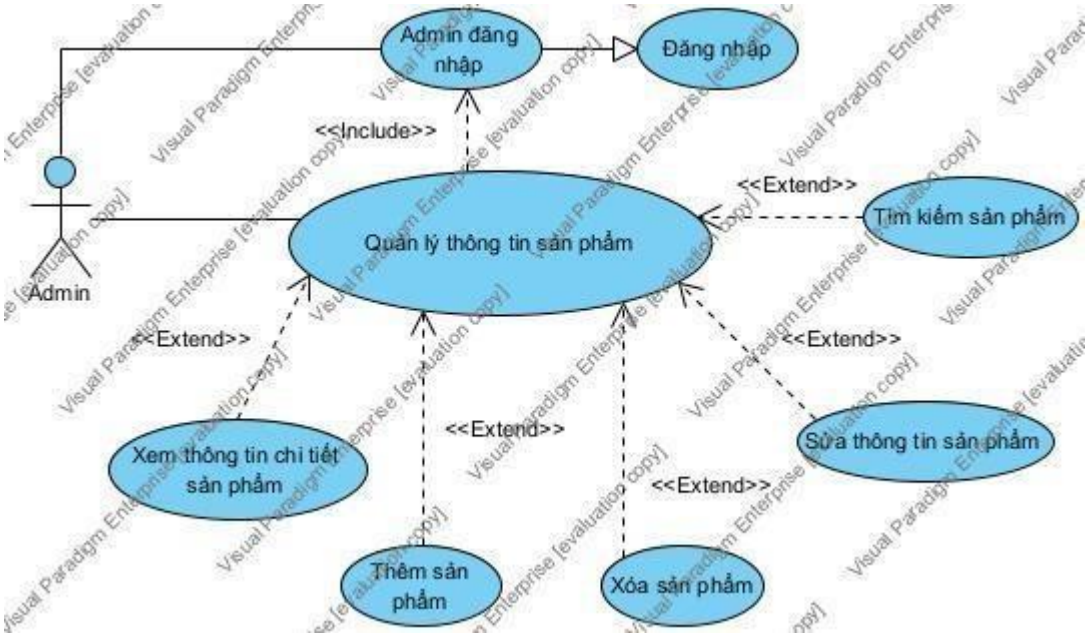
2.2.7. Thanh toán



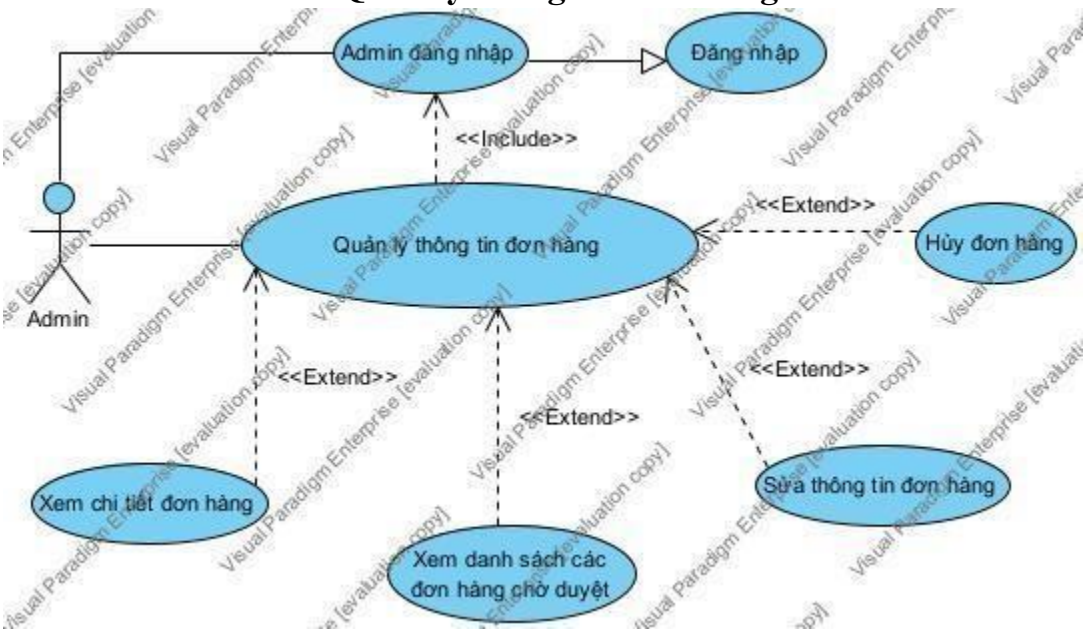
2.2.8. Xem thông tin giao dịch



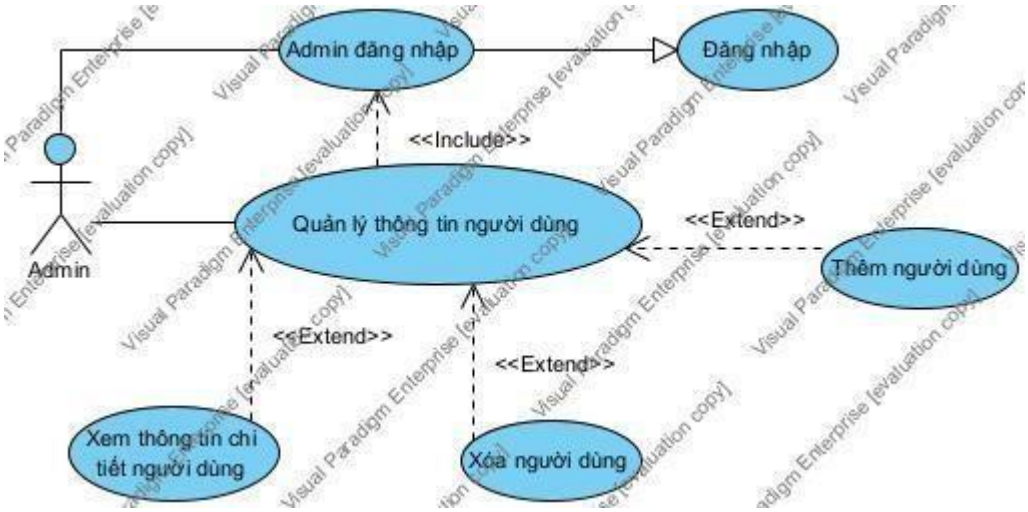
2.2.9. Quản lý thông tin sản phẩm



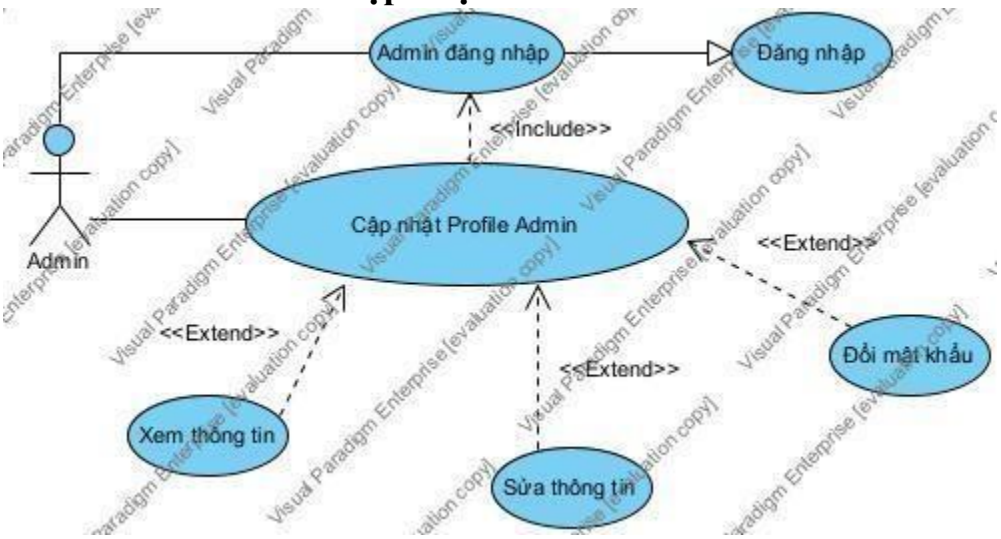
2.2.10. Quản lý thông tin đơn hàng



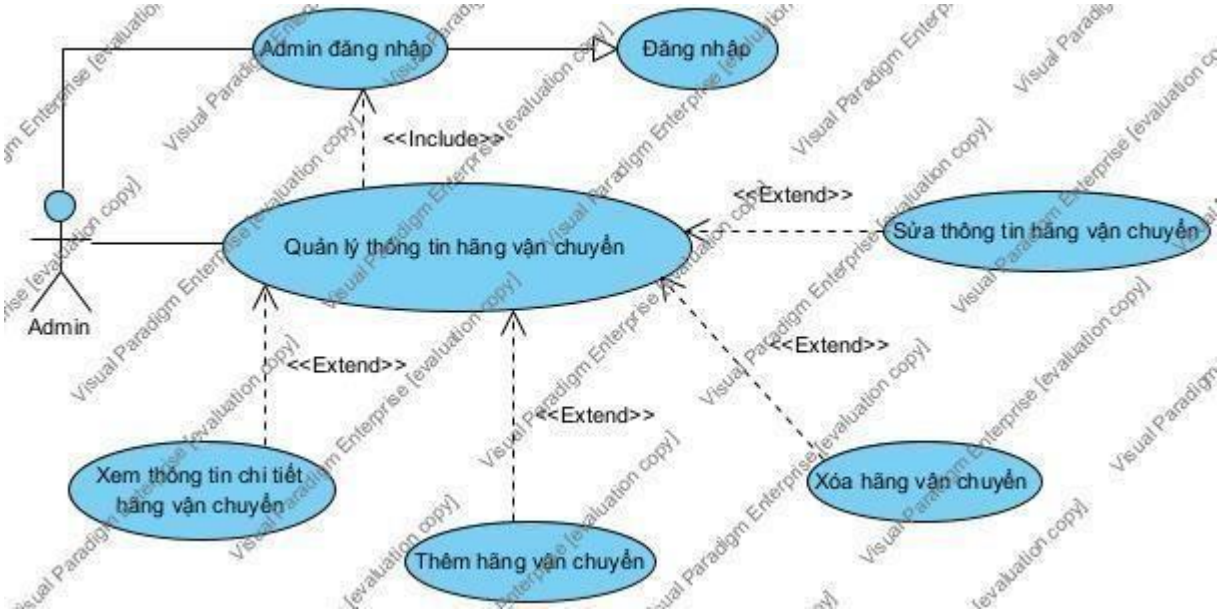
2.2.11. Quản lý thông tin người dùng



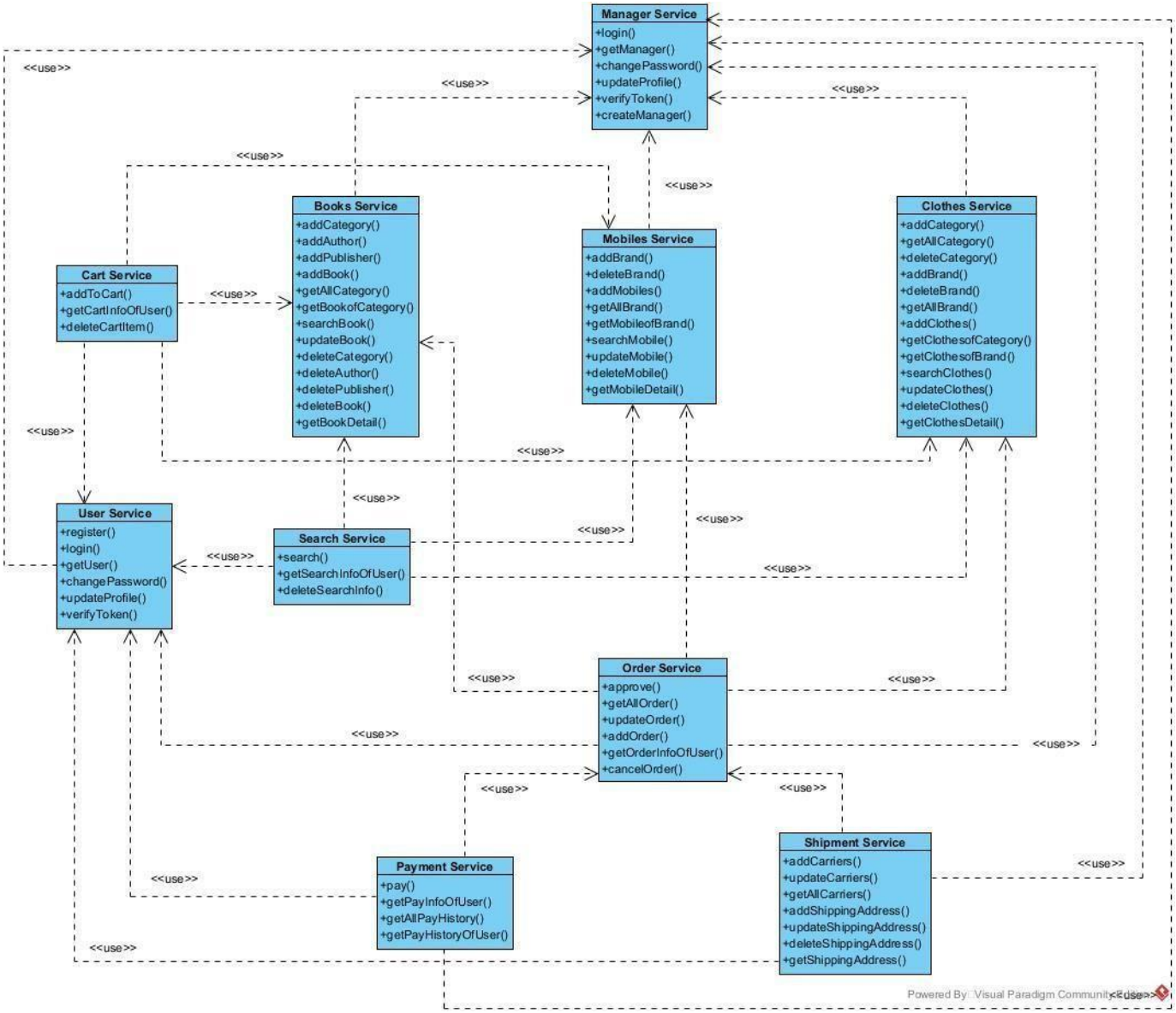
2.2.12. Cập nhật Profile Admin



2.2.13. Quản lý thông tin hãng vận chuyển



3. Vẽ biểu đồ phân rã Hệ ecomSys thành các service và các tương tác giữa các dịch vụ



4. Trình bày sử dụng các dạng communication giữa các service với code cho hệ ecomSys

Trong hệ ecomSys, hiện em đang sử dụng giao tiếp đồng bộ giữa các service:

- Giữa Manager Service và Books Service: Books Service yêu cầu xác thực từ Manager Service để quản lý Book

```
class AddBookView(APIView):
    def post(self, request):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            serializer = BookSerializer(data=request.data, context={'request': request})
            if serializer.is_valid():
                serializer.save()
                return Response(serializer.data, status=status.HTTP_201_CREATED)
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

- Giữa Manager Service và Mobiles Service: Mobiles Service yêu cầu xác thực từ Manager Service để quản lý Mobiles

```
class AddMobileView(APIView):
    def post(self, request):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            serializer = MobileSerializer(data=request.data, context={'request': request})
            if serializer.is_valid():
                serializer.save()
                return Response(serializer.data, status=status.HTTP_201_CREATED)
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

- Giữa User Service và Search Service: Search Service yêu cầu xác thực từ User Service để xem/xóa thông tìm kiếm của user

```
class ShowSearchView(APIView):
    def get(self, request):

        token_verification_url = "http://localhost:4000/api/ecomSys/user/info/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)
        if response.status_code == 200:
            user_id = response.json().get('id')
            searchs_instance = Search.objects.filter(is_active__in=[True], user_id=user_id).all()
            serializer = SearchSerializer(searchs_instance, many=[True])
            return Response(serializer.data, status=status.HTTP_200_OK)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)

class DeleteSearchView(APIView):
    def delete(self, request, key):
        token_verification_url = "http://localhost:4000/api/ecomSys/user/info/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            user_id = response.json().get('id')

            try:
                search = Search.objects.get(user_id=user_id, key=key, is_active__in=[True])
            except Search.DoesNotExist:
                return Response({'error': 'Search not found'}, status=status.HTTP_404_NOT_FOUND)

            serializer = SearchSerializer()
            serializer.destroy(search)

            return Response({'message': 'Search soft deleted'}, status=status.HTTP_204_NO_CONTENT)

        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

- Giữa User Service và Cart Service: Cart Service yêu cầu xác thực từ User Service để quản lý Cart

```python
class AddToCartView(APIView):
    def post(self, request):

        token_verification_url = "http://localhost:4000/api/ecomSys/user/info"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            user_id = response.json().get('id')
            product_id = request.data.get('product_id')
            cart_item = CartItem.objects.filter(is_active=True, user_id=user_id, product_id=product_id).first()
            if cart_item:
                serializer = UpdateCartItemSerializer(instance=cart_item, data=request.data)
                if serializer.is_valid():
                    serializer.save()
                    return Response(serializer.data, status=status.HTTP_200_OK)
                return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
            else:
                request.data['user_id'] = user_id
                serializer = CartItemSerializer(data=request.data)
                if serializer.is_valid():
                    serializer.save()
                    return Response(serializer.data, status=status.HTTP_201_CREATED)
                return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

- Giữa Search Service và Books Service, Mobiles Service: Search Service gửi yêu cầu tìm kiếm đến Books Service và Mobiles Service để nhận về kết quả

```python
class SearchView(APIView):
    def post(self, request):
        key = request.query_params.get('key', '')

        token_verification_url = "http://localhost:4000/api/ecomSys/user/info/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            user_id = response.json().get('id')
            Search.objects.create(key=key, user_id=user_id)

        result = []
        result += self.search_book(key)
        result += self.search_mobile(key)

        return Response(result, status=status.HTTP_200_OK)

    def search_book(self, key):
        book_service_url = "http://localhost:4002/api/ecomSys/book/search/{}/".format(key)

        book_response = requests.get(book_service_url)
        if book_response.status_code == 200:
            return book_response.json()
        return []

    def search_mobile(self, key):
        mobile_service_url = "http://localhost:4005/api/ecomSys/mobile/search/{}/".format(key)

        mobile_response = requests.get(mobile_service_url)
        if mobile_response.status_code == 200:
            return mobile_response.json()
        return []

    def search_clothes(self, key):
        clothes_service_url = "http://localhost:4006/api/ecomSys/clothes/search/{}/".format(key)

        # Call API to Clothes Service
        return []
```

- Giữa Cart Service và Books Service, Mobiles Service: Cart Service gửi yêu cầu lấy thông tin sản phẩm đến Books Service và Mobiles

```python
class CartView(APIView):
    def get(self, request):
        token_verification_url = "http://localhost:4000/api/ecomSys/user/info"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            user_id = response.json().get('id')
            cart_items = CartItem.objects.filter(is_active=True, user_id=user_id)
            cart_total = 0
            cart_item_data = []

            for cart_item in cart_items:
                product = self.get_product(cart_item.type, cart_item.product_id)
                if product:
                    cart_item_data.append({
                        'quantity': cart_item.quantity,
                        'product': product,
                        'total': cart_item.quantity * product.get('price', 0) * (100-product.get('sale', 0))/100
                    })
                    cart_total += cart_item.quantity * product.get('price', 0 ) * (100-product.get('sale', 0))/100
            response_data = {
                'cart_items': cart_item_data,
                'cart_total': cart_total
            }
            return Response(response_data, status=status.HTTP_200_OK)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)

    def get_product(self, type, product_id):
        if type == 'book':
            product_url = "http://localhost:4002/api/ecomSys/book/detail/{}/".format(product_id)
        if type == 'mobile':
            product_url = "http://localhost:4002/api/ecomSys/mobile/detail/{}/".format(product_id)
        response = requests.get(product_url)

        if response.status_code == 200:
            return response.json()
        return None
```

Service để lấy sản phẩm

II. BÀI TẬP 4

Những packages cần thiết:

sgiref==3.7.2

harset-normalizer==3.3.2

j angorestframework==3.15. e

3.6

dna==3.6

sq1c1ient==2.2.4

ymongo==3.12.3

requests==2.31. e

2.4

zdata==2Q4.1

Danh sách port cho từng service:

```
. env\Scripts\activate.bat
. env\Scripts\activate.bat
. env\Scripts\activate.bat
. \. env\Scripts\activate.bat
. \. env\Scripts\activate.bat
. env\Scripts\activate.bat
. env\Scripts\activate.bat
. \. env\Scripts\activate.bat
. \. env\Scripts\activate.bat
. \. env\Scripts\activate.bat
. \. env\Scripts\activate.bat
. bat &&
. \user service\manage.py runserver 8000
. \order\manage.py runserver 8001
. \clothes service\manage.py runserver 8002
. \shipment_service\manage.py runserver 8003
```

book_service:

model book:

```python
from django.db import models

# Create your models here.

class Book(models.Model):
    name = models.CharField(max_length=200)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    author = models.CharField(max_length=200)

    def __str__(self):
        return self.name
```

Serializer:

```python
from rest_framework import serializers
from .models import Book

class BooksSerializer(serializers.ModelSerializer):
    name = serializers.CharField(max_length=200, required=True)
    price = serializers.DecimalField(max_digits=10, decimal_places=2, required=True)
    author = serializers.CharField(max_length=200, required=True)

    class Meta:
        model = Book
        fields = ('__all__')
```

URL:

```python
from django.urls import path
from .views import BookView

urlpatterns = [
    path('', BookView.as_view()),
    path('<int:id>/', BookView.as_view()),
    # path('<int:id>/update/', BookView.as_view()),
]
```

Views.py:

```python
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .models import Book
from .serializers import BooksSerializer
from django.shortcuts import get_object_or_404
# Create your views here.

class BookView(APIView):

    def get(self, request, id=None):
        if id != None:
            result = Book.objects.get(id=id)
            serializers = BooksSerializer(result)
            return Response({'status': 'success',"message":"Query book success", "data":serializers.data}

        result = Book.objects.all()
        serializers = BooksSerializer(result, many=True)
        return Response({'status': 'success',"message":"Query book success", "data":serializers.data}, st

    def post(self, request, id=None):
        serializer = BooksSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response({"status": "success","message":"book added", "data": serializer.data}, status
        else:
            return Response({"status": "error","message":serializer.errors,"data":""}, status=status.HTTP_

    def patch(self, request, id=None):
        if("id" in request.data):
            id=request.data["id"]
        result = Book.objects.get(id=id)
        result.price=float(str(result.price))
        serializer = BooksSerializer(result, data = request.data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response({"status": "success","message":"books updated", "data": serializer.data})
        else:
            return Response({"status": "error","message":"update books error", "data": serializer.errors}

    def delete(self, request, id=None):
        result = get_object_or_404(Book, id=id)
        result.delete()
        return Response({"status": "success", "message": "Book deleted", "data":""})
```

Clothes_service:

Model clothes:

```python
from django.db import models

# Create your models here.

class Clothes(models.Model):
    name = models.CharField(max_length=200)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    author = models.CharField(max_length=200)

    def __str__(self):
        return self.name
```

Serializer:

```python
from rest_framework import serializers
from .models import Clothes

class ClothesSerializer(serializers.ModelSerializer):
    name = serializers.CharField(max_length=200, required=True)
    price = serializers.DecimalField(max_digits=10, decimal_places=2, required=True)
    author = serializers.CharField(max_length=200, required=True)

    class Meta:
        model = Clothes
        fields = ('__all__')
```

URL:

```python
from django.urls import path
from .views import ClothesView

urlpatterns = [
    path('', ClothesView.as_view()),
    path('<int:id>/', ClothesView.as_view()),
    # path('<int:id>/update/', BookView.as_view()),
]
```

VIEW:

```python
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .models import Clothes
from .serializers import ClothesSerializer
from django.shortcuts import get_object_or_404
# Create your views here.

class ClothesView(APIView):

    def get(self, request, id=None):
        if id != None:
            result = Clothes.objects.get(id=id)
            serializers = ClothesSerializer(result)
            return Response({'status': 'success',"message":"Query clothes success", "data":serializers.da

        result = Clothes.objects.all()
        serializers = ClothesSerializer(result, many=True)
        return Response({'status': 'success',"message":"Query clothes success", "data":serializers.data},

    def post(self, request, id=None):
        serializer = ClothesSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response({"status": "success","message":"clothes added", "data": serializer.data}, sta
        else:
            return Response({"status": "error","message":serializer.errors,"data":""}, status=status.HTTP_

    def patch(self, request, id=None):
        if("id" in request.data):
            id=request.data["id"]
        result = Clothes.objects.get(id=id)
        result.price=float(str(result.price))
        serializer = ClothesSerializer(result, data = request.data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response({"status": "success","message":"clothes updated", "data": serializer.data})
        else:
            return Response({"status": "error","message":"update clothes error", "data": serializer.error

    def delete(self, request, id=None):
        result = get_object_or_404(Clothes, id=id)
        result.delete()
        return Response({"status": "success", "message": "clothes deleted", "data":""})
```

Mobile_service:

Model:

```python
from django.db import models

# Create your models here.

class Mobile(models.Model):
    name = models.CharField(max_length=200)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    author = models.CharField(max_length=200)

    def __str__(self):
        return self.name
```

Serializer:

```python
from rest_framework import serializers
from .models import Mobile

class MobileSerializer(serializers.ModelSerializer):
    name = serializers.CharField(max_length=200, required=True)
    price = serializers.DecimalField(max_digits=10, decimal_places=2, required=True)
    author = serializers.CharField(max_length=200, required=True)

    class Meta:
        model = Mobile
        fields = ('__all__')
```

URL:

```python
from django.urls import path
from .views import MobileView

urlpatterns = [
    path('', MobileView.as_view()),
    path('<int:id>/', MobileView.as_view()),
    # path('<int:id>/update/', MobileView.as_view()),
]
```

VIEW:

```python
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .models import Mobile
from .serializers import MobileSerializer
from django.shortcuts import get_object_or_404
# Create your views here.

class MobileView(APIView):

    def get(self, request, id=None):
        if id != None:
            result = Mobile.objects.get(id=id)
            serializers = MobileSerializer(result)
            return Response({'status': 'success',"message":"Query mobile success", "data":serializers.dat

        result = Mobile.objects.all()
        serializers = MobileSerializer(result, many=True)
        return Response({'status': 'success',"message":"Query mobile success", "data":serializers.data},

    def post(self, request, id=None):
        serializer = MobileSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response({"status": "success","message":"mobile added", "data": serializer.data}, stat
        else:
            return Response({"status": "error","message":serializer.errors,"data":""}, status=status.HTTP_

    def patch(self, request, id=None):
        if("id" in request.data):
            id=request.data["id"]
        result = Mobile.objects.get(id=id)
        result.price=float(str(result.price))
        serializer = MobileSerializer(result, data = request.data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response({"status": "success","message":"mobile updated", "data": serializer.data})
        else:
            return Response({"status": "error","message":"update mobile error", "data": serializer.errors

    def delete(self, request, id=None):
        result = get_object_or_404(Mobile, id=id)
        result.delete()
        return Response({"status": "success", "message": "mobile deleted", "data":""})
```

User_service:

Model:

```python
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
from django.db import models

# This is our model for user registration.


class User(models.Model):
    # The following are the fields of our table.
    fname = models.CharField(max_length=50)
    lname = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    mobile = models.CharField(max_length=12)
    password = models.CharField(max_length=50)
    address = models.CharField(max_length=200)
    # It will help to print the values.

    def __str__(self):
        return '%s %s %s %s %s %s' % (self.fname, self.lname, self.email, self.mobile, self.password, sel

```

Serializer:

```python
from rest_framework import serializers
from .models import User

class UserSerializer(serializers.ModelSerializer):
    fname = serializers.CharField(max_length=50)
    lname = serializers.CharField(max_length=50)
    email = serializers.CharField(max_length=50)
    mobile = serializers.CharField(max_length=12)
    password = serializers.CharField(max_length=50)
    address = serializers.CharField(max_length=200)

    class Meta:
        model = User
        fields = ('__all__')
```

URL:

```python
from .import views
from django.urls import path
from .views import UserRegister, UserLogin

urlpatterns = [
    path('api/userregistration/', UserRegister.as_view()),
    path('api/userlogin/', UserLogin.as_view())
]

```

VIEW:

```python
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .models import User
from .serializers import UserSerializer
from django.shortcuts import get_object_or_404
# Create your views here.

def user_validation(uname, password):
    user_data = User.objects.filter(
        email=uname, password=password).first()
    if user_data:
        return user_data
    else:
        return None

class UserRegister(APIView):
    def post(self, request):
        serializer = UserSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response({"status": "success","message":"Register success", "data": serializer.data},
        else:
            return Response({"status": "error","message":serializer.errors,"data":""}, status=status.HTTP_
```

```python
class UserLogin(APIView):
    def post(self, request):
        result = user_validation(request.data["email"], request.data["password"])
        if(result):
            print(result)
            serializer = UserSerializer(result)
            return Response({"status": "success","message":"Login success", "data": serializer.data}, sta
        else:
            return Response({"status": "fail","message":"Login fail", "data": "Username or Password is wr
```

Shipment_service:

Model:

```python
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
from django.db import models
# Create your models here.


class Shipment(models.Model):
    # The following are the fields of our table.
    fname = models.CharField(max_length=50)
    lname = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    mobile = models.CharField(max_length=12)
    address = models.CharField(max_length=200)
    product_type = models.CharField(max_length=200)
    product_id = models.IntegerField()
    quantity = models.CharField(max_length=5)
    payment_status = models.CharField(max_length=15)
    transaction_id = models.CharField(max_length=5)
    shipment_status = models.CharField(max_length=20)

    # It will help to print the values.
    def __str__(self):
        return '%s %s %s %s %s %s %s %s %s %s' % (self.fname, self.lname, self.email, self.mobile, self.p
```

Serializer:

```python
from rest_framework import serializers
from .models import Shipment
import requests
import json

class ShipmentSerializer(serializers.ModelSerializer):
    fname = serializers.CharField(max_length=50)
    lname = serializers.CharField(max_length=50)
    email = serializers.CharField(max_length=50)
    mobile = serializers.CharField(max_length=12)
    address = serializers.CharField(max_length=200)
    product_type = serializers.CharField(max_length=200)
    product_id = serializers.IntegerField()
    product = serializers.SerializerMethodField()
    quantity = serializers.CharField(max_length=5)
    payment_status = serializers.CharField(max_length=15)
    transaction_id = serializers.CharField(max_length=5)
    shipment_status = serializers.CharField(max_length=20)

    class Meta:
        model = Shipment
        fields = ('__all__')
    def get_product(self, obj):
        try:
            if(obj.product_type=="book"):
                product_resp = requests.get("http://localhost:8005/api/book/"+str(obj.product_id)+"/?format=json",headers={'Accept': 'ap
            elif (obj.product_type=="mobile"):
                product_resp = requests.get("http://localhost:8006/api/mobile/"+str(obj.product_id)+"/?format=json",headers={'Accept': '
            elif (obj.product_type=="clothes"):
                product_resp = requests.get("http://localhost:8007/api/clothes/"+str(obj.product_id)+"/?format=json",headers={'Accept':

            return product_resp.json()["data"]
        except Exception as ex:
            print(ex)
            return None
```

View:

```python
from .import views
from django.urls import path
from .views import shipment_status, shipment_reg_update

urlpatterns = [
    path('api/shipment_status/', shipment_status.as_view()),
    path('api/shipment_reg_update/', shipment_reg_update.as_view())
]

```

View:

```python
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .models import Shipment
from .serializers import ShipmentSerializer
from django.shortcuts import get_object_or_404
import requests
# Create your views here.

class shipment_status(APIView):
    def post(self, request):
        result = Shipment.objects.filter(email=request.data["email"])
        serializers = ShipmentSerializer(result, many=True)
        return Response({"status": "success","message":"Query shipment status success", "data": serializers.data}, status=status.HTTP_20

class shipment_reg_update(APIView):
    def post(self, request):
        serializer = ShipmentSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response({"status": "success","message":"Shipment Register/Update success", "data": serializer.data}, status=status.H
        else:
            return Response({"status": "error","message":serializer.errors,"data":""}, status=status.HTTP_400_BAD_REQUEST)
```

Sử dụng API shipment_status:s